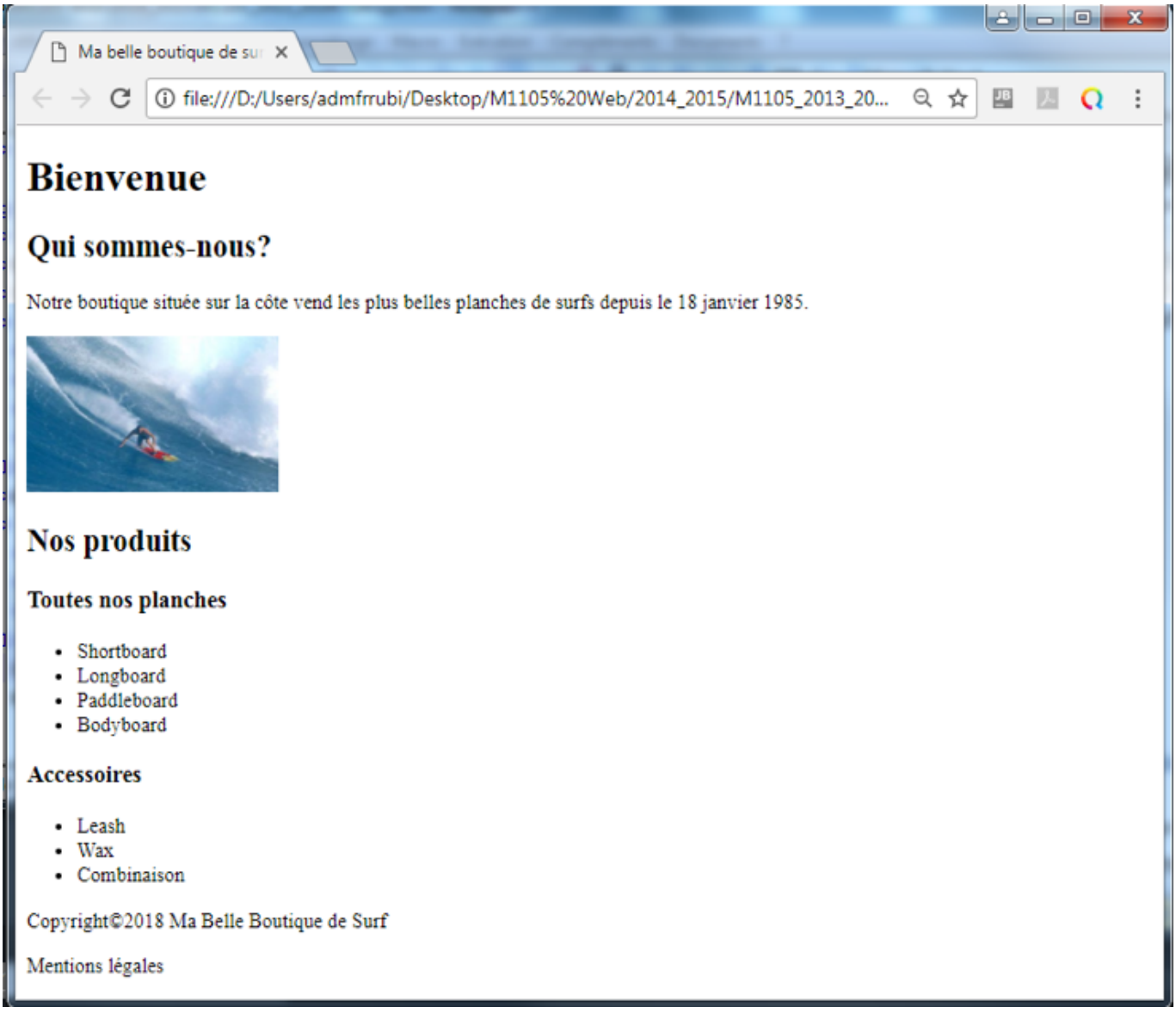
Introduction à CSS

Réalisation des pages pour le site de la boutique de surf

Question 1

En prenant exemple sur l'image ci-dessous construisez, dans un fichier index.html, le squelette de votre site contenant :

- Un titre de page.
- Ajoutez un titre principal, suivi de titres secondaires.
- Un petit paragraphe d'introduction expliquant aux visiteurs qu'ils sont sur le site d'une boutique de planches de surfs serait utile.
- Une illustration
- Ajoutez une liste pour les différentes planches et accessoires disponibles.
- Un paragraphe « Copyright » et un paragraphe « Mentions légales »

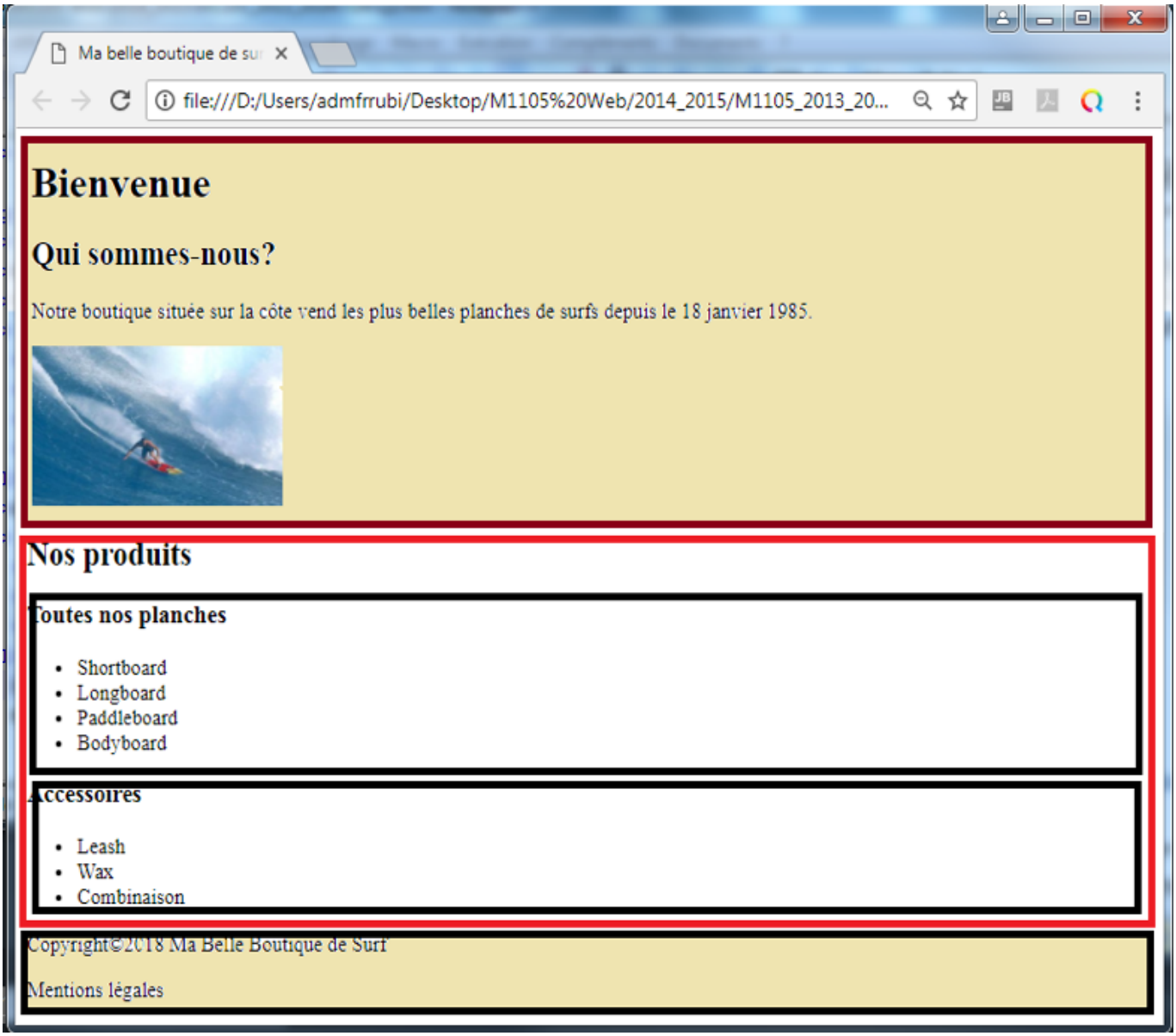


Question 2 : zoning

Nous venons d'ajouter des paragraphes, des titres, des listes, mais il serait intéressant de **rassembler les éléments par zones** notamment dans le but de pouvoir, par la suite, leur donner un style particulier.

Découper la page en une zone « en-tête », une zone « produits » et une zone « pied de page ». La zone « produits » sera redécoupé en deux sous zones. Vous utiliserez les balises sémantiques HTML5 adéquates et si nécessaire la balise div.

La balise `div` permet d'englober tout autre type d'élément (à la différence d'un paragraphe qui ne peut être composé d'autres paragraphes par exemple).
On décide de regrouper dans l'en-tête les éléments allant de "Bienvenue" à l'image du surfeur.



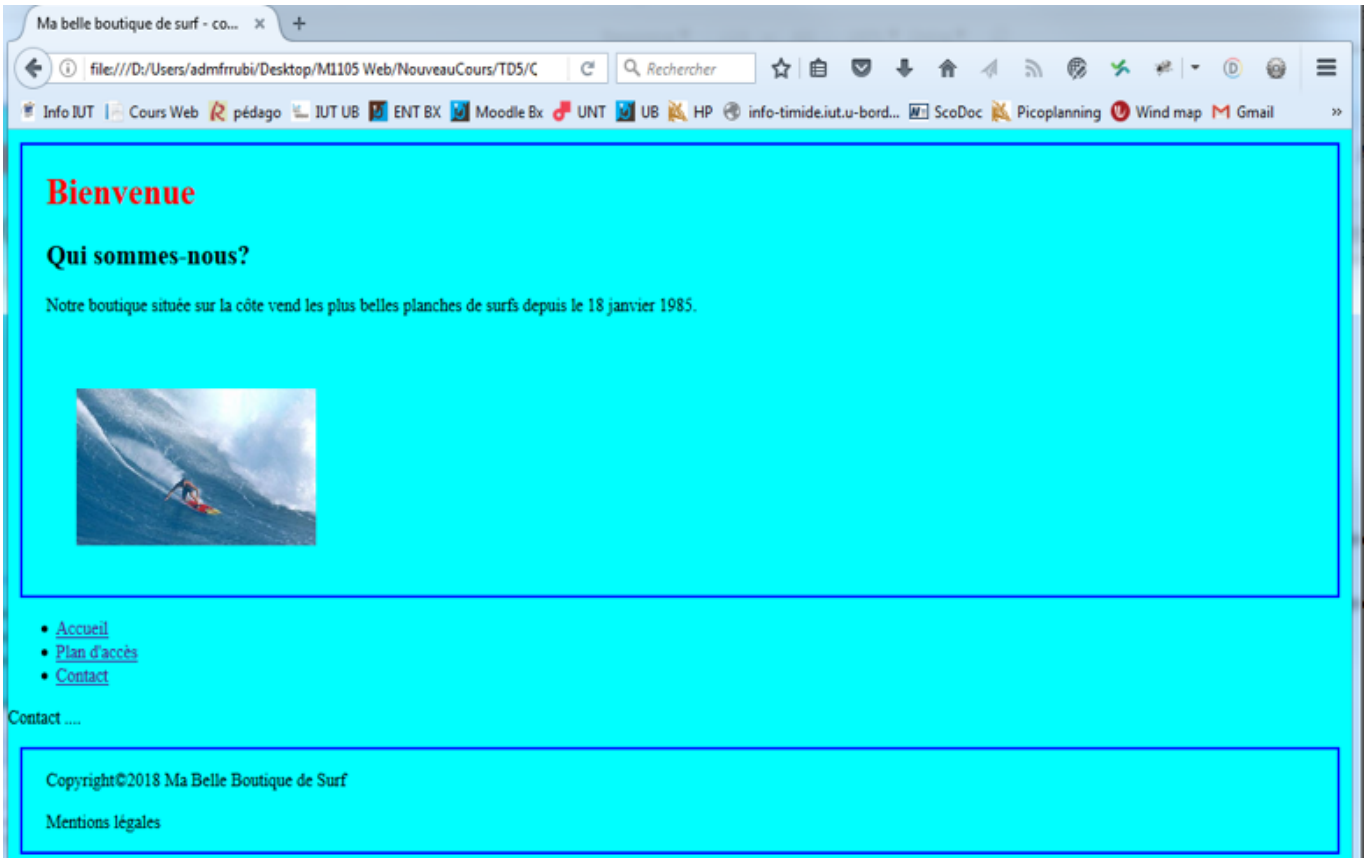
Question 2 : navigation dans le site

Notre site est composé de plusieurs pages ; nous allons donc créer d'autres pages et des liens permettant de naviguer d'une page à l'autre.

1. En vous basant sur la première page réalisée (les zones « en-tête » et « pied de page » seront reproduits à l'identique), créez deux nouvelles pages :
    - une page « plan d'accès »

- une page « contact ».

2. Ajoutez une zone « menu » permettant de naviguer de page en page.



Modification du style de la page

`CSS` (**Cascading Style Sheets**) est un langage qui permet de changer l'apparence des éléments d'une page (la taille, le style, la couleur d'un texte, la couleur de fond, les bordures,la position d'un élément dans la page...).
On peut rajouter des styles CSS de plusieurs manières :

- Chaque élément HTML possède un attribut `style` dans lequel nous pouvons ajouter des déclarations.

```
<div style="background-color:orange; border:1px solid black; color:yellow;
font-size:150%; padding:1em;"> Cette balise div a du style !</div>
```

- La balise `<style>` existe également, dans laquelle nous pouvons regrouper plusieurs déclarations dans le `head`

```
<style type="text/css">
  div {
    background-color:#339;
    color:#fff;
    padding:15px;
    border-bottom:5px solid red;
    margin-bottom:15px;
```

```
        }
</style>
```

- Inclure un fichier contenant tout ou partie des déclarations (inclusion de feuille de style) placée dans la partie en-tête (`<head>`) de la page.

```
<link href="style.css" rel="stylesheet"/>
```

La troisième solution apporte le plus d'avantages, car elle permet de **regrouper en un endroit unique un ensemble de déclarations**. Il est alors aisé de changer un style s'appliquant sur plusieurs pages sans avoir à modifier chacune des pages.

Une déclaration CSS comporte un sélecteur suivi d'une liste de couples « propriété valeur »:

> sélecteur { propriété1 : valeur1 ; propriété2 : valeur2 ; … }

par exemple pour les paragraphes :

```
p {
  display: block;
  margin-top: 1em;
  margin-bottom: 1em;
}
```

Comment appliquer un style aux balises HTML standards

Avec les CSS, vous pouvez changer la présentation de toutes les balises HTML standards. Il vous suffit de spécifier le nom de la balise et de faire figurer vos définitions comme suit :

Cette définition de style est à placer dans une feuille de styles ou dans la section `<head>` de votre page.

```
p {
  font-weight:bold;
  line-height:1.3em;
}
```

Ce style s'appliquera à toutes les balises `p` de la page.

Comment utiliser des classes pour appliquer un style

Vous pouvez attribuer à chaque élément HTML une ou plusieurs `classes`. C'est vous qui définirez le nom de ces classes et qui déciderez de leurs styles.

Les styles définis dans les classes remplaceront les styles "normaux" des éléments auxquels ils s'appliquent.

Pour créer une classe, vous devez simplement faire figurer son nom précédé **d'un point**. Pour éviter toute ambiguïté, votre nom de classe ne doit pas comporter d'espace.

Cette définition de style est à placer dans une feuille de styles ou dans la section `<head>` de votre page.

```
.mon-style {
  color:red;
}
```

Pour appliquer le style défini dans votre classe à un élément, ajouter la mention **class="nom-du style"** dans la définition de la balise :

```
<p class="mon-style">Le style s'applique à ce paragraphe </p>
```

Les éléments HTML peuvent avoir plusieurs classes

Pour appliquer plusieurs classes au même élément, précisez simplement la liste de classes en séparant leurs noms par un espace :

```
.mon-style1 {
  color:yellow;
}
.mon-style2 {
  background-color:#A0A0A0;
  font-weight:bold;
}
```

```
<p class="mon-style1 mon-style2">Les styles des deux classes s'appliquent
à ce paragraphe</p>
```

Comment utiliser des "ID" pour appliquer un style

Les éléments HTML peuvent se voir attribuer un **ID** (identification) en plus ou à la place d'une classe.

Le principe de l'ID est très similaire à celui de la classe à une exception prés :

Plusieurs éléments peuvent avoir la même classe.
Il ne doit y avoir qu'un seul élément ayant un ID donné.

Pour créer un ID, vous devez faire figurer son nom précédé d'un **dièse #**. Pour éviter toute ambiguïté, votre nom d'ID ne doit pas comporter d'espace.

```
#mon-style {
  color:red;
}
```

Pour appliquer le style défini dans votre ID à un élément, ajouter la mention **id="nom-du style"** comme attribut de la balise :

```
<p id="mon-style">Le style s'applique à ce paragraphe </p>
```

Question 3 : Un peu de style svp !

1. Créer une feuille de style permettant de définir :

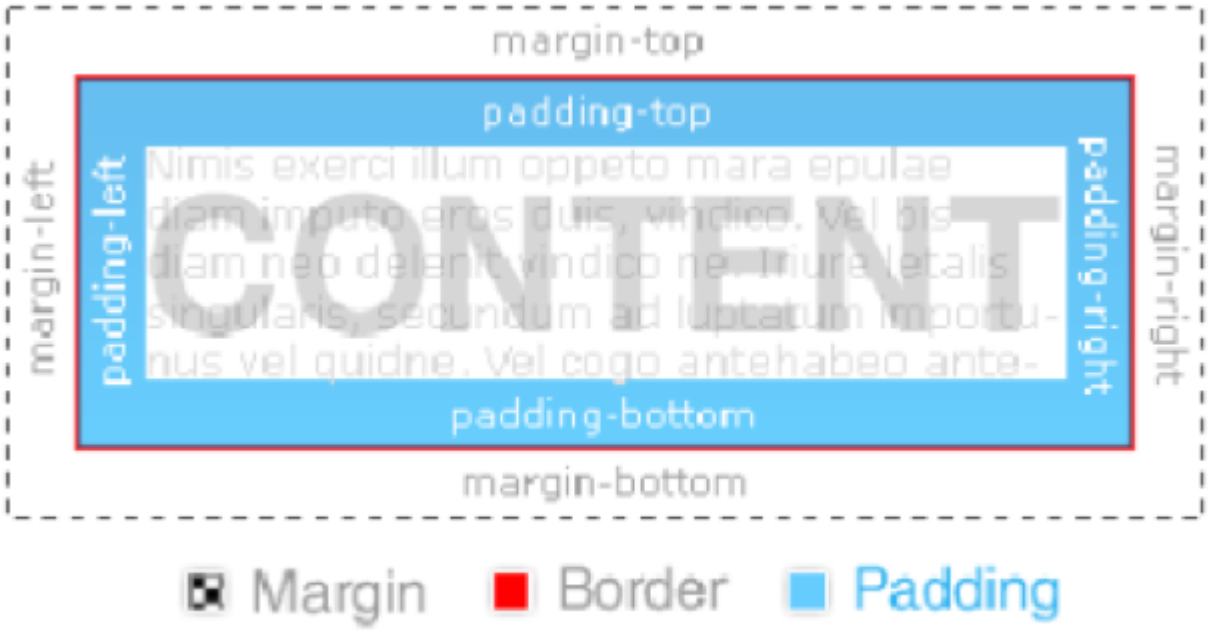
- La couleur du fond du site (`background-color`), valeur « aqua » ou #00ffff
- La couleur du titre de la page (`color`), valeur « red » ou #ff0000
- La marge de l'image (`margin`), marge haute : 40px, marge basse : 20px, marge gauche : 25px

2. Ajoutez dans vos pages html la balise link mettant en relation votre feuille de style.

Marge extérieure et marge intérieure

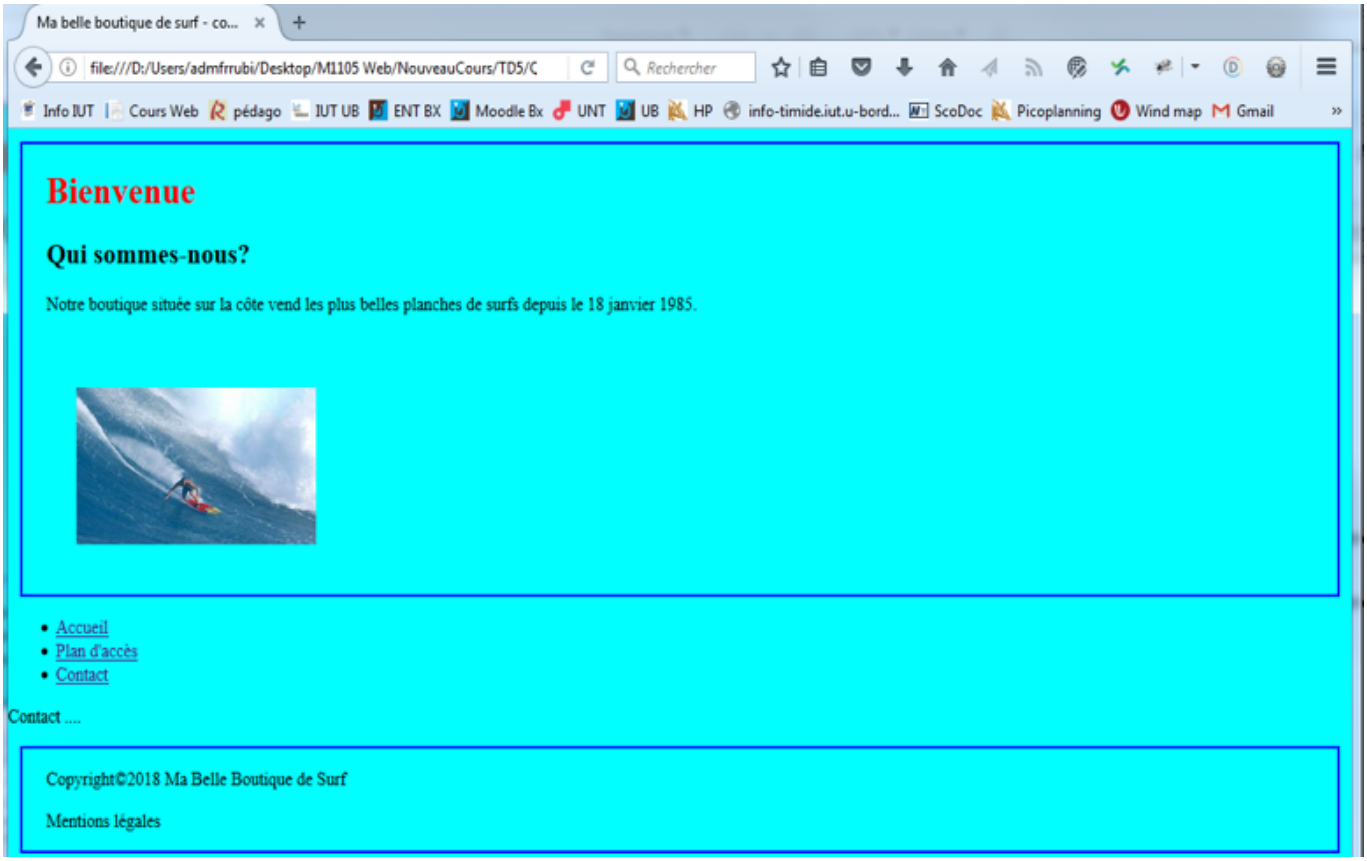
Un élément peut posséder deux types de marges de part et d'autre de la bordure.

- Des marges extérieures (margin)
- des marges intérieures (padding).



Question 4 : bordure, marge class et id !

1. Sur la partie haute (header) du site, ajoutez une bordure de 2px, solide et de la couleur de votre choix, une marge intérieure à gauche et à droite de 20px, une marge extérieure de 10 px.
2. Sur la partie basse du site (footer), ajoutez une bordure de 2px, solide et de la couleur de votre choix, une marge intérieure à gauche et à droite de 20px, une marge extérieure de 10px.
3. Tous les éléments de la liste du “menu“ ne sont plus précédés d’un point,
4. Tous les éléments des listes de produits ou d’accessoires soient précédés d’un carré et plus d’un point.



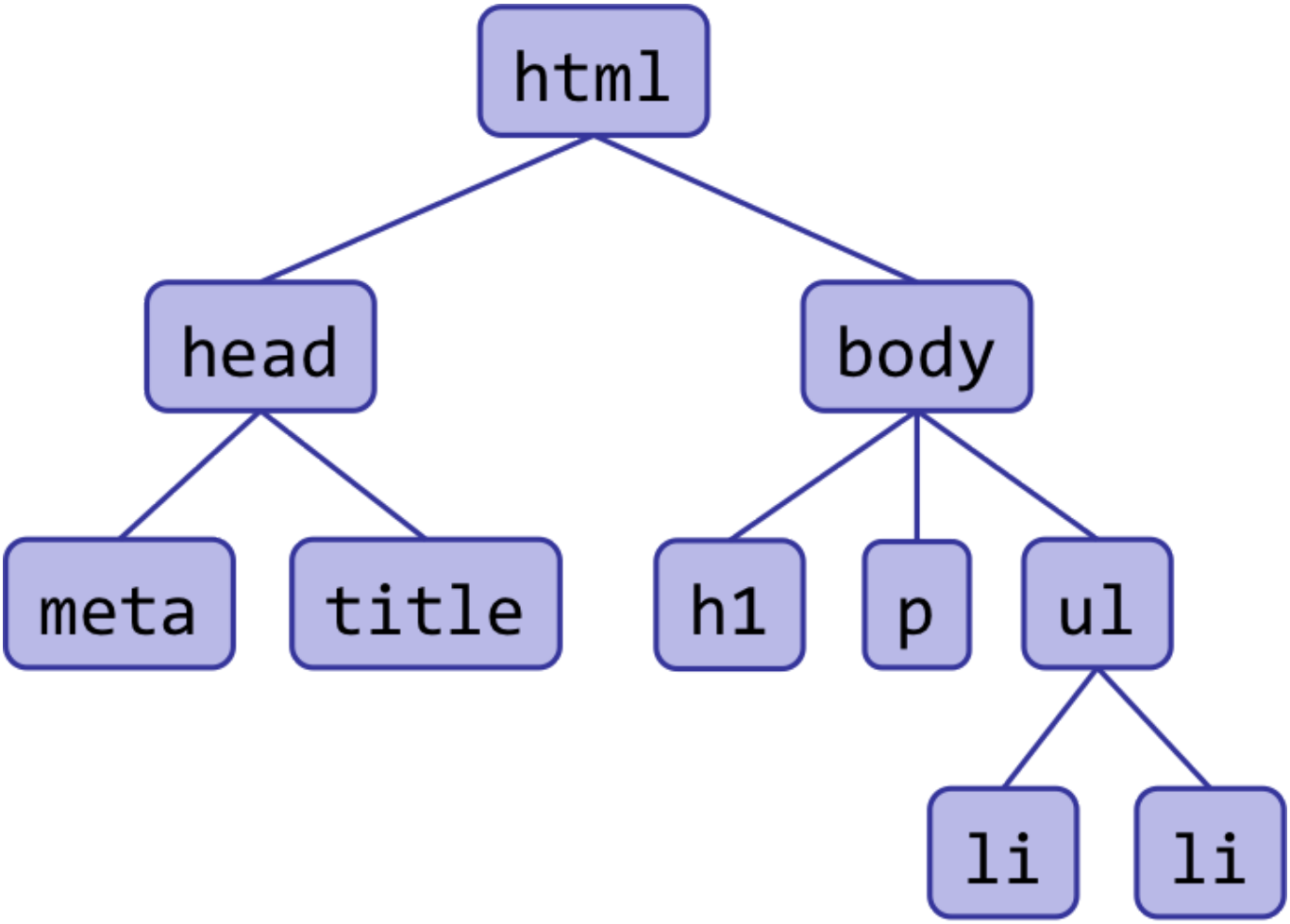
`

Sélection en héritage

Le problème avec les classes, c'est qu'elles peuvent vite devenir très nombreuses, et "polluer" les fichiers HTML. Pour résoudre ce problème, CSS permet de sélectionner uniquement les « descendants » d'un élément (à noter que l'on peut restreindre en ne sélectionnant que les enfants « directs »).

Le modèle Parent-Enfant(s)

L'héritage des CSS est fondé sur le modèle *Parent-Enfant(s)* : chaque élément Enfant reçoit en héritage tous les styles de son élément Parent.



Par exemple, la balise `<html>` est parent de `<body>`, et `<table>` est parent de `<tr>` qui lui-même est parent de `<td>`.

Cet héritage est très pratique et permet d'éviter beaucoup de répétitions inutiles : en effet, en attribuant une propriété à un parent (par exemple font-size: 1.5em), elle sera transmise à tous ses enfants, mais aussi aux enfants de ces enfants, etc...

Précision : l'élément enfant héritera de toutes les propriétés de l'élément parent uniquement si ces propriétés s'héritent, car l'héritage ne fonctionne pas non plus sur toutes les propriétés css (margin, padding et autres propriétés de bloc que nous verrons ensuite)

Le **selecteur de descendance**, représenté par un **blanc** permet de combiner deux sélecteurs afin de cibler les éléments qui correspondent au second sélecteur uniquement si ceux-ci ont un élément ancêtre qui correspond au premier sélecteur.

```
/* Les éléments <li> qui sont des descendants */
/* d'un <ul class="truc"> */
ul.truc li {
  margin: 2em;
}
```

Le **selecteur d'enfants X > Y** (ou de descendance directe)

```
#container > ul {
border: 1px solid black;
}
```

La différence entre le X Y standard et X > Y est que ce dernier ne **sélectionnera que des enfants directs**. Par exemple, soit le balisage suivant :

```
<div id="container">
   <ul>
      <li> List Item
        <ul>                -> non ciblé
           <li> Child </li>
        </ul>
      </li>
      <li> List Item </li>
      <li> List Item </li>
      <li> List Item </li>
   </ul>
</div>
```

Le sélecteur #container > ul ne ciblera que les uls qui sont enfants directs de la div avec l'id container. A l'inverse, il ne ciblera pas le ul qui est enfant du premier li.

Question 5 : heritage

En utilisant la sélection en héritage, faire que les titres des listes de produits soient en italique de couleur #395699 et avec une marge intérieure gauche de 5 pixels.

Premiers pas vers le positionnement des éléments dans la page

Il existe plusieurs schémas de positionnement en CSS : le schéma de positionnement dans **le Flux** (positionnement par défaut), et de nombreux autres schémas de positionnement : **absolu, fixé, flottant, relatif, tabulaire, en grille, à l'aide des boites flexibles, …**

Chaque schéma de positionnement a ses règles et spécificités. Mieux vaut bien comprendre ces règles pour éviter des interactions ou des comportements non sollicités... et déclarer que les CSS ne fonctionnent pas ! 😃.

Avec l'émergence des smartphones, des tablettes tactiles, des grands écrans, en tenant compte de la problématique liée à **l'accessibilité** (par exemple : s'assurer que les éléments positionnés ne masquent pas d'autre contenu sur la page lorsqu'on zoome afin d'augmenter la taille du texte, règles WCAG), on souhaite aujourd'hui que l'affichage de la page soit réalisée dynamiquement en fonction des caractéristiques de la zone d'affichage (responsive design)

Notion de flux

Le **flux** d'un document est le comportement **naturel d'affichage des éléments** d'une page web. Les éléments se succèdent dans l'ordre où ils sont déclarés dans le code HTML.

Les éléments de type `block` (`h1`, `p`, `ul`, `ol`, `dl`, `table`, `blockquote`, etc.) Les éléments de type `inline` (`a`, `img`, `strong`, `abbr`, etc.) Un élément de type block peut s'identifier à un "bloc" d'informations qu'on va pouvoir manipuler aisément. Il se différencie des éléments en ligne sur différents points, dont ceux-ci :

- Il occupe toute la largeur de son conteneur
- Il permet l'attribution de marges verticales
- Il permet la modification de sa hauteur et largeur

Tout élément peut être "reclassé" dans la famille opposée grâce à la propriété `display`.

La propriété position : lorsque le flux ne suffit plus

La position relative

La position relative (`position:relative`) permet de décaler un élément par rapport à la position qu'il avait dans le flux.
Les éléments qui le suivent et le précèdent ne sont pas influencés par ce décalage puisqu'ils considèrent que l'élément est toujours dans le flux à sa position initiale. **En pratique, ce comportement est rarement recherché bien qu'il puisse s'avérer utile dans certains cas**.

Attribuer à un élément une position relative peut par contre être pratique, voire indispensable, dans d'autres situations:

- Servir de référent à un élément enfant positionné en absolu.(cf ci-dessous)
- Bénéficier de la possibilité d'utiliser la propriété z-index pour gérer des superpositions d'éléments (propriété inopérante pour des éléments du flux)

La position absolue

La position absolue (`position:absolute`) permet de ne pas dépendre de l'ordre dans lequel les éléments HTML sont déclarés dans le code, contrairement aux flottants que nous verrons plus tard.

> Il faut voir le positionnement absolu comme étant une méthode qui retire totalement un élément du flux: il n'existe pour ainsi dire plus aux yeux des éléments qui, eux, restent dans le flux.

Un élément positionné en absolu **se réfère** non pas à son parent direct, mais **au premier ancêtre positionné qu'il rencontre**.

Il est capital de noter qu'un élément bénéficiant d'une position absolue ne bougera pas de sa position initiale tant que l'une des propriétés `top`, `bottom`, `left` ou `right` n'a pas été précisée; il s'agit d'ailleurs là d'un comportement appliquable à toutes les positions.

La position fixe

Le positionnement fixe (`position:fixed`) s'apparente au positionnement absolu, à l'exception des points suivants:

- Lorsque le positionnement est précisé (top, right, …), l'élément est **toujours positionné par rapport à la fenêtre du navigateur**

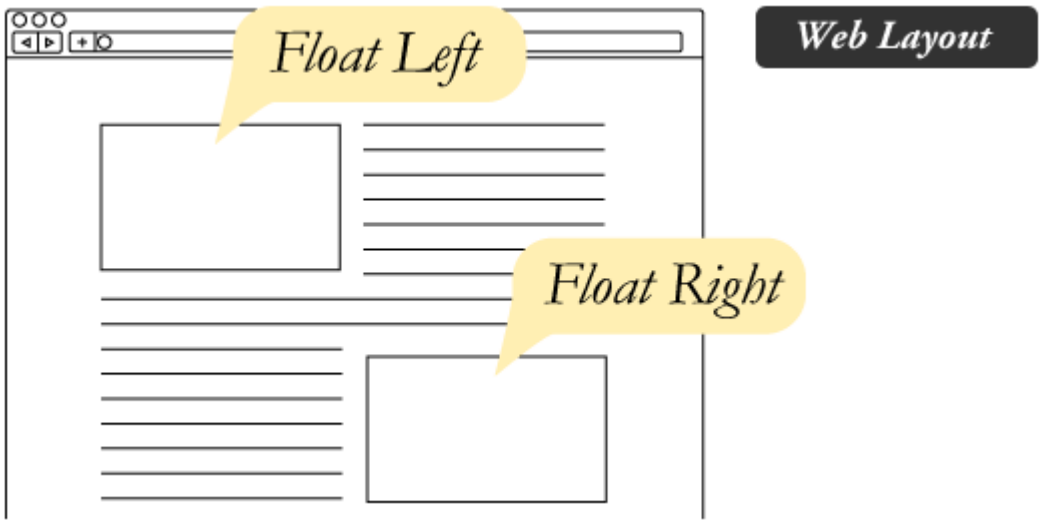
- L'élément est fixé à un endroit et ne pourra se mouvoir, même lors de la présence d'une barre de défilement.Il ne bouge plus de la position initialement définie.

La position statique

La position statique (`position:static`) correspond à la valeur par défaut d'un élément du flux. On ne l'indique pas sauf pour remettre un élément dans le flux.

la position float

La propriété float existe pour répondre à un besoin typographique précis: la création d'habillages.

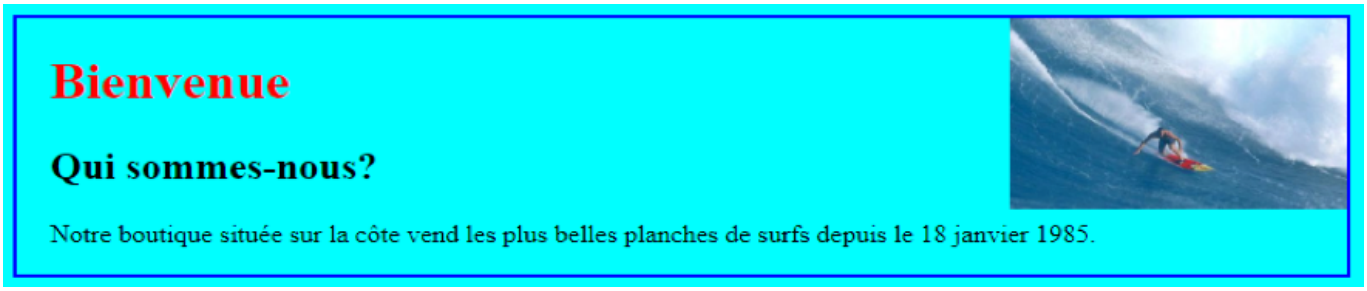


Un élément flottant est ôté partiellement du flux et placé à l'extrême gauche (`float:left`) ou droite (`float:right`) de son conteneur, forçant par la même occasion tout contenu du flux qui suit à l'envelopper. Deux objets flottants dans la même direction se rangeront côte à côte, seul un contenu demeuré dans le flux qui les succède immédiatement initiera l'habillage.

Nettoyer les flottants La propriété `clear` s'utilise conjoitement aux float et permet à un élément de ne plus subir le comportement d'habillage et de se caler en-dessous de ce dernier.
Le `clear` autorise un nettoyage des flottants exclusivement à gauche (`clear:left`), à droite (`clear:right`) ou les deux simultanément (`clear:both`).

Question 6 : positionnement

1. En utilisant le positionnement relatif et le positionnement absolu, faire que l'image se positionne dans le coin supérieur droit de l'en-tête en déclarant :
    - la zone d'en-tête en positionnement relatif,
    - l'image (sans marges) en positionnement absolu avec une distance à zéro du haut et une distance à zéro du bord droit du bloc référent.
2. Que peut-il se passer lorsque la zone d'affichage diminue en largeur ?



decouverte de flex

Le **Flexbox Layout** (Boîte flexible) vise à fournir un moyen plus efficace de disposer, d'aligner et de répartir l'espace entre les éléments d'un conteneur, même lorsque leur taille est inconnue et / ou dynamique (en fait "flexible").

L'idée principale derrière la disposition flex est de donner au conteneur la possibilité de modifier la largeur / hauteur (et l'ordre) de ses enfants afin de remplir au mieux l'espace disponible (principalement pour s'adapter à tout type de dispositifs d'affichage et de tailles d'écran).

Un conteneur flexible étend les éléments pour remplir l'espace disponible ou les réduit pour éviter tout débordement.

propriété du conteneur

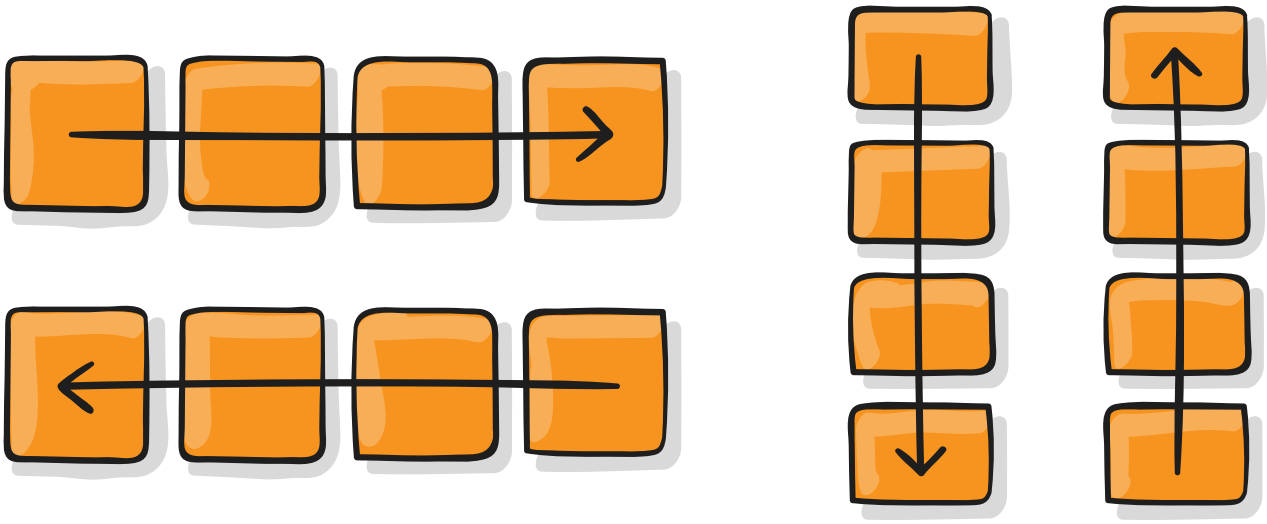
display Ceci définit un conteneur flex;

```
.container {
  display: flex;
}
```

flex-direction

Ceci établit l'axe principal, définissant ainsi la direction. (row par defaut)

```
.container {
  flex-direction: row | row-reverse | column | column-reverse;
}
```



flex-wrap Par défaut, les éléments flex tenteront tous de s'adapter à une seule ligne. Vous pouvez modifier cela et permettre aux éléments d'être renvoyés à la ligne avec cette propriété.

```
.container{
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

- nowrap (par défaut): tous les éléments flexibles seront sur une seule ligne
- wrap: les éléments flexibles se disposent sur plusieurs lignes, de haut en bas.
- wrap-reverse: les éléments flexibles se disposent sur plusieurs lignes de bas en haut.

exemple de flex-wrap à voir ici

flex-flow Il s'agit d'un raccourci pour les propriétés flex-direction et flex-wrap

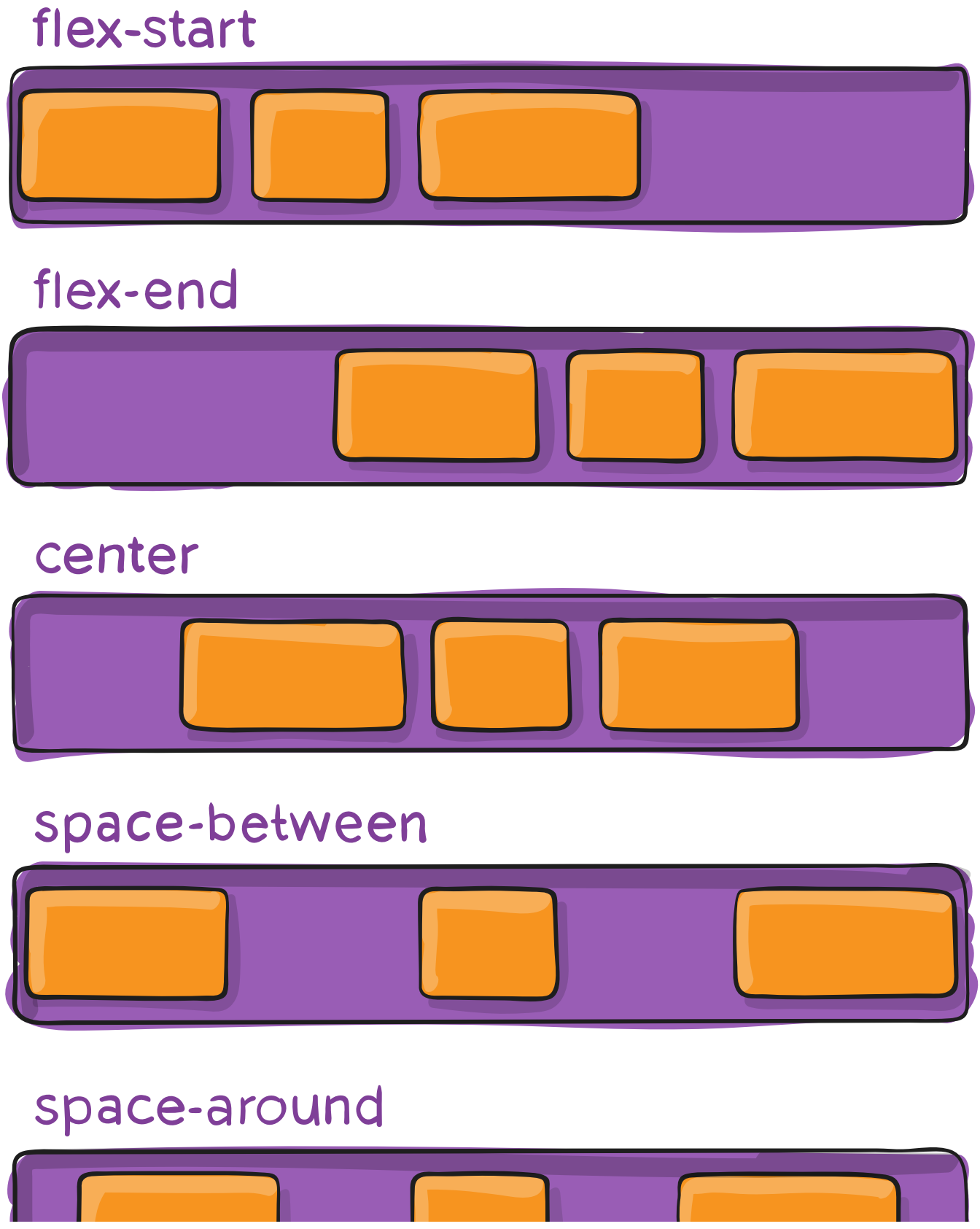
```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

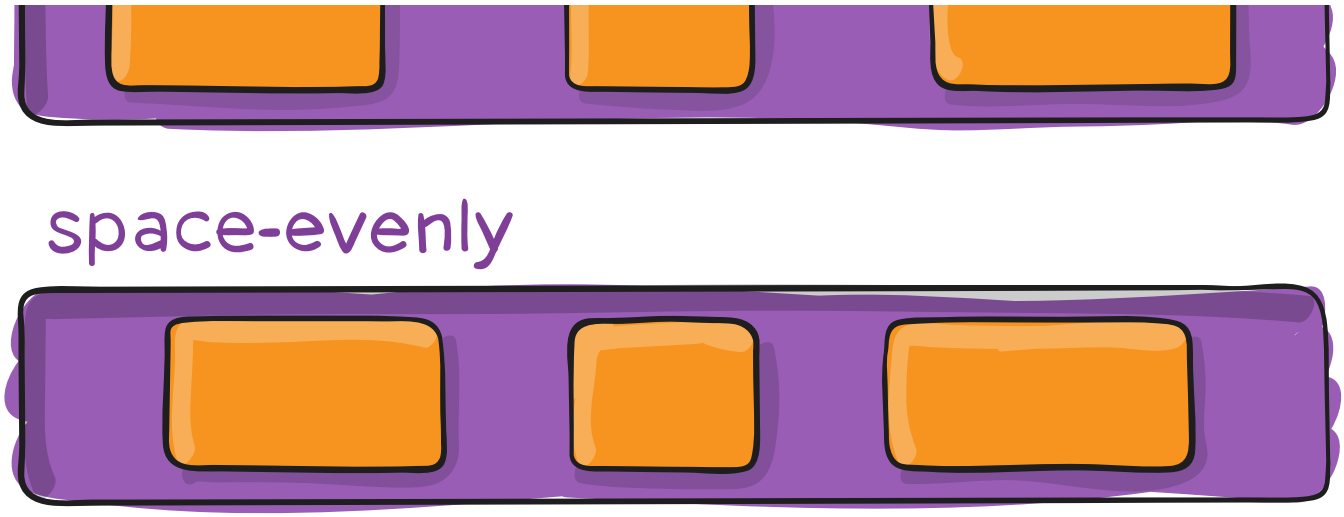
justify-content Ceci définit l'alignement le long de l'axe principal. Cela permet de répartir un espace libre supplémentaire lorsque tous les éléments flexibles d'une ligne sont inflexibles ou flexibles, mais ont atteint leur taille maximale. Il exerce également un certain contrôle sur l'alignement des éléments lorsqu'ils débordent de la ligne.

```
.container {
  justify-content: flex-start | flex-end | center | space-between | space-
around | space-evenly;
}
```

`flex-start` (valeur par défaut): les éléments sont regroupés vers la ligne de départ `flex-end`: les éléments sont emballés vers la ligne de fond `center`: les éléments sont centrés le long de la ligne `space-between`: les

éléments sont répartis uniformément dans la ligne; le premier élément est sur la ligne de départ, le dernier élément sur la ligne de fin `space-around`: les éléments sont répartis uniformément dans la ligne avec un espace égal autour d'eux. Notez que visuellement, les espaces ne sont pas égaux, car tous les éléments ont un espace égal des deux côtés. Le premier élément aura une unité d'espace contre le bord du conteneur, mais deux unités d'espace entre l'élément suivant, car cet élément a son propre espacement qui s'applique. `space-evenly`: les éléments sont répartis de sorte que l'espacement entre deux éléments quelconques (et l'espace sur les bords) soit égal.



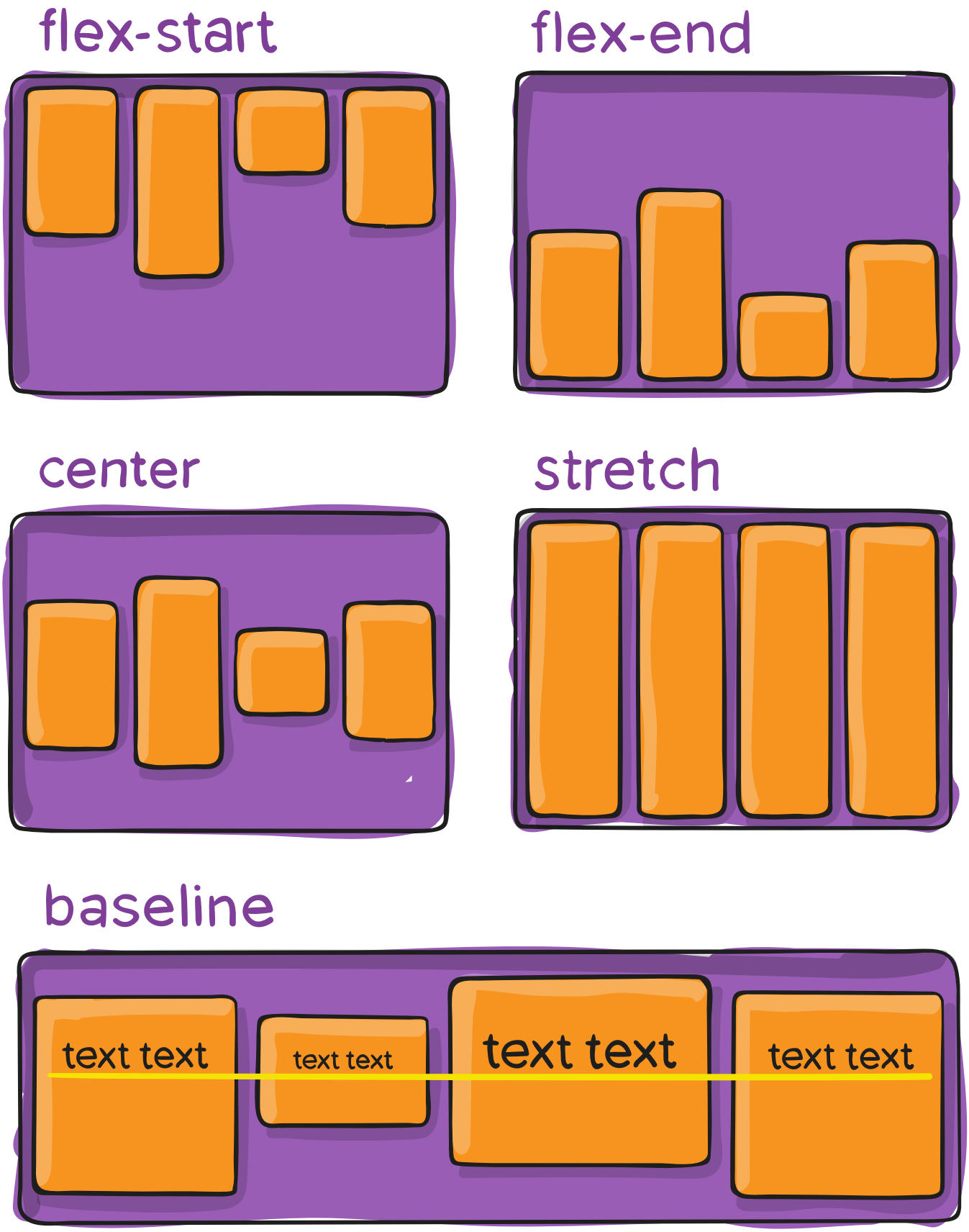


align-items

Ceci définit le comportement par défaut de la manière dont les éléments flexibles sont disposés le long de l'axe secondaire sur la ligne en cours.

```
.container {
  align-items: stretch | flex-start | flex-end | center | baseline;
}
```

`stretch` (par défaut): étirer pour remplir le conteneur (toujours respecter min-width / max-width) `flex-start`: le bord de la marge de départ croisé des éléments est placé sur la ligne de départ croisée `flex-end`: le bord de la marge transversale des éléments est placé sur la ligne transversale `center`: les articles sont centrés dans l'axe transversal `baseline`: les éléments sont alignés tels que leurs lignes de base alignées

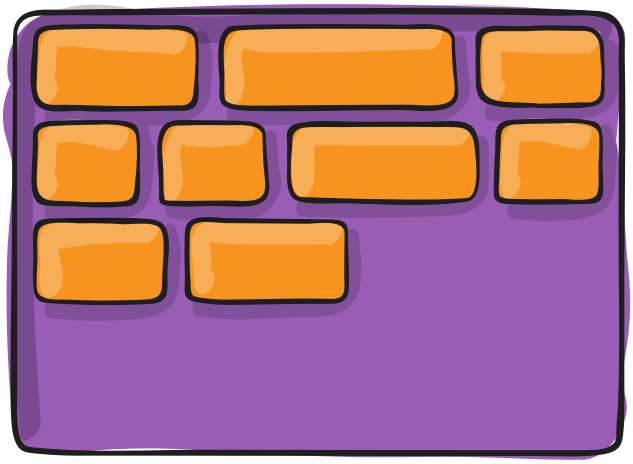


align-content

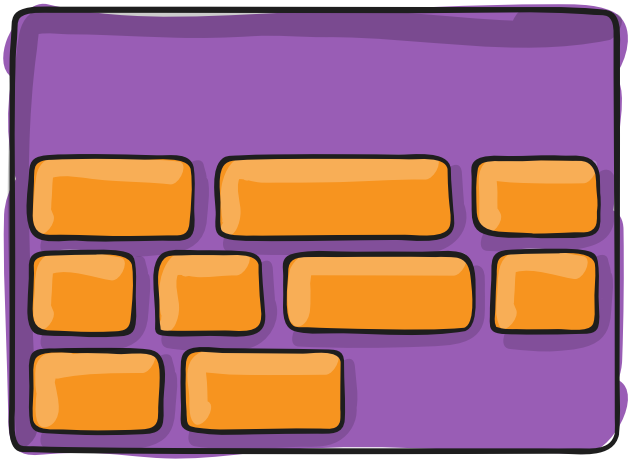
```
.container {
  align-content: flex-start | flex-end | center | space-between | space-
```

```
around | stretch;
}
```

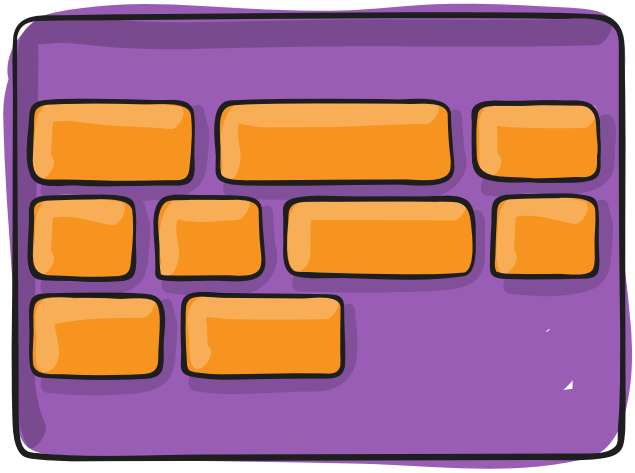
flex-start



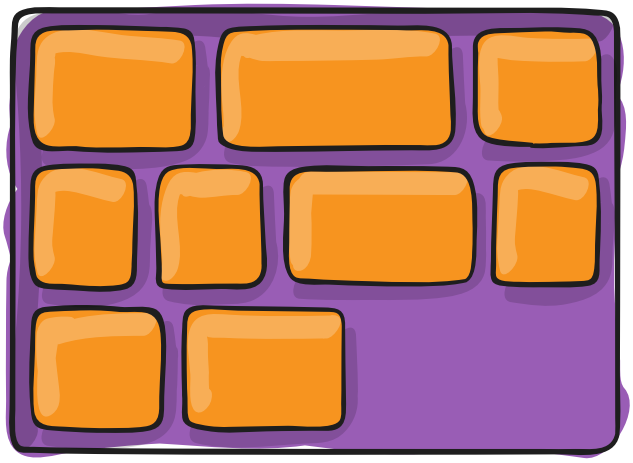
flex-end



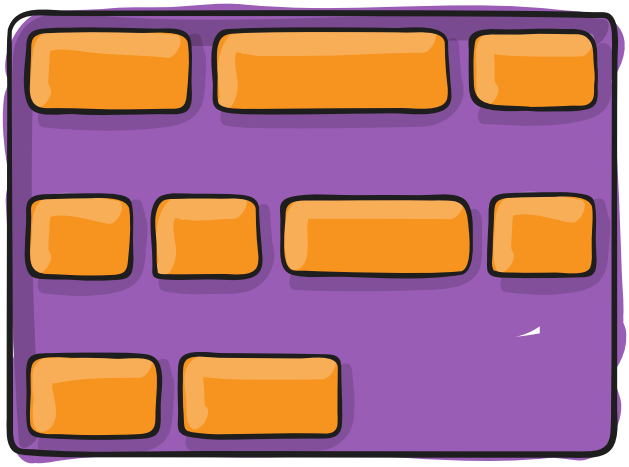
center



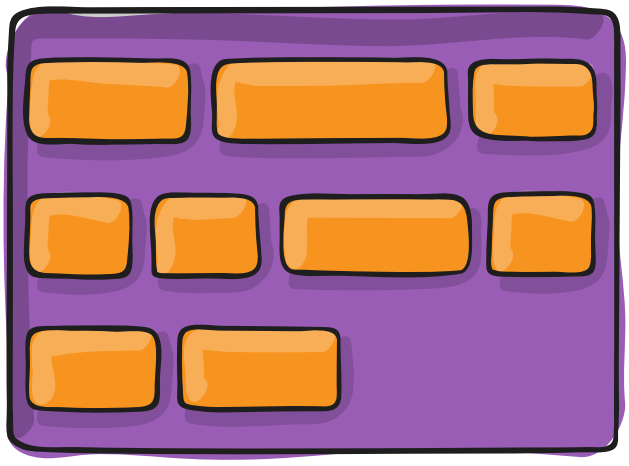
stretch



space-between



space-around



Propriétés pour les enfants (éléments flexibles)

order

Par défaut, les éléments flex sont disposés dans l'ordre source. Cependant, cette propriété contrôle l'ordre dans lequel elles apparaissent dans le conteneur flex. exemple ici

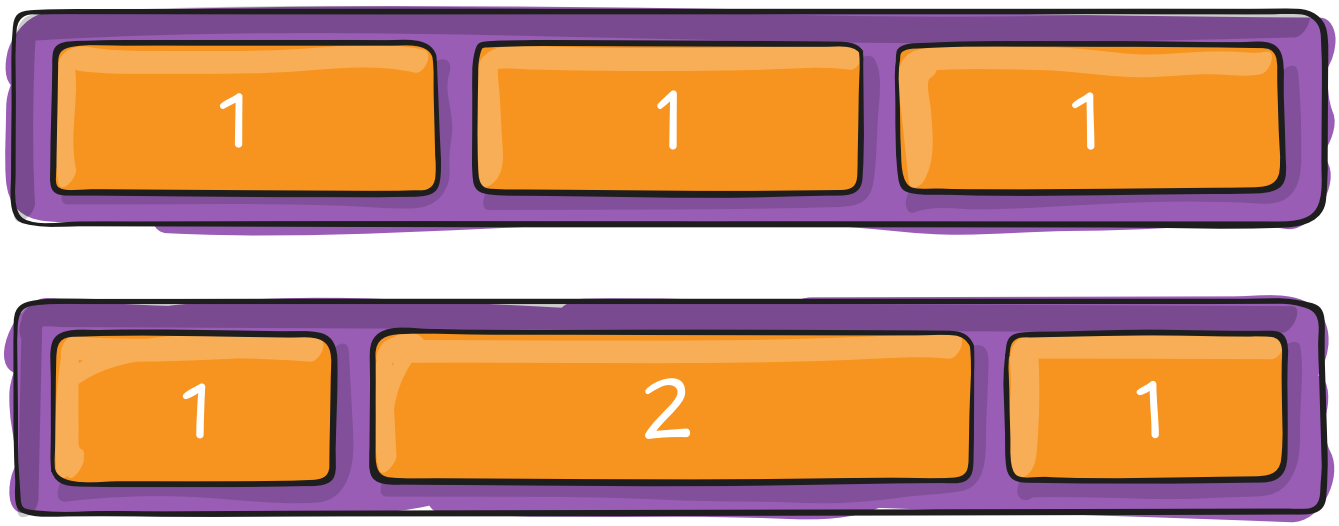
```
.item {
  order: <integer>; /* default is 0 */
}
```

Flex-Grow

La propriété CSS `flex-grow` définit le facteur d'expansion d'un élément flexible. Elle indique la quantité d'espace que l'élément devrait consommer dans un conteneur flexible relativement à la taille des autres éléments du même conteneur. Si tous les éléments voisins possèdent le même facteur d'expansion, ils recevront tous la même part d'espace.

La plupart du temps flex-grow est utilisé avec les autres propriétés flexibles flex-shrink et flex-basis. On pourra utiliser la propriété raccourcie flex afin de s'assurer que toutes les valeurs des propriétés flexibles auront été définies.

```
.item {
  flex-grow: <number>; /* default 0 */
}
```



exemple ici

`flex-shrink` Ceci définit la possibilité pour un élément flexible de se contracter si nécessaire.

```
.item {
  flex-shrink: <number>; /* default 1 */
}
```

flex-basis Ceci définit la taille par défaut d'un élément avant que l'espace restant ne soit distribué. Ce peut être une longueur (par exemple 20%, 5rem, etc.) ou un mot clé. Le mot clé ***auto*** signifie "regardez ma propriété width ou height"

flex Ceci est le raccourci pour flex-grow, flex-shrink et flex-basis combiné. Les deuxième et troisième paramètres (flex-shrink et flex-basis) sont facultatifs. La valeur par défaut est 0 1 auto.

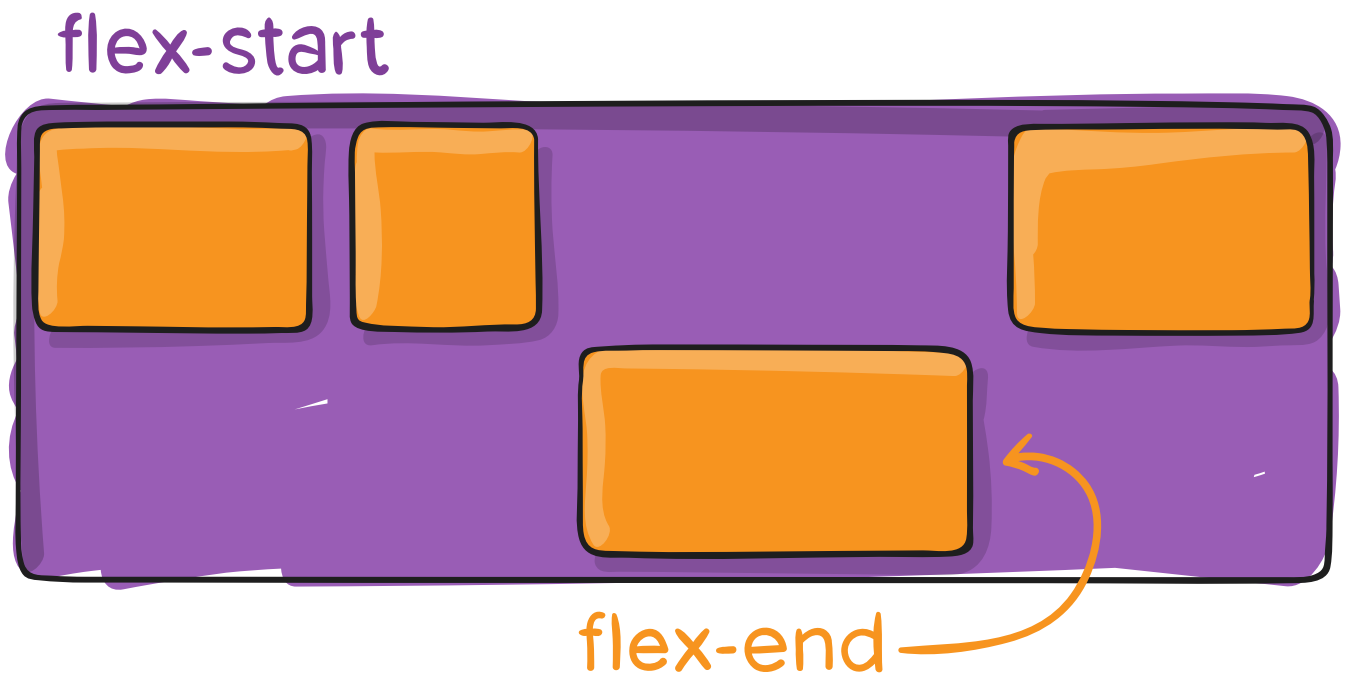
```
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

Il est recommandé d'utiliser cette propriété abrégée plutôt que de définir les propriétés individuelles.

align-self

Cela permet de remplacer l'alignement par défaut (ou celui spécifié par align-items) pour des éléments flexibles individuels.

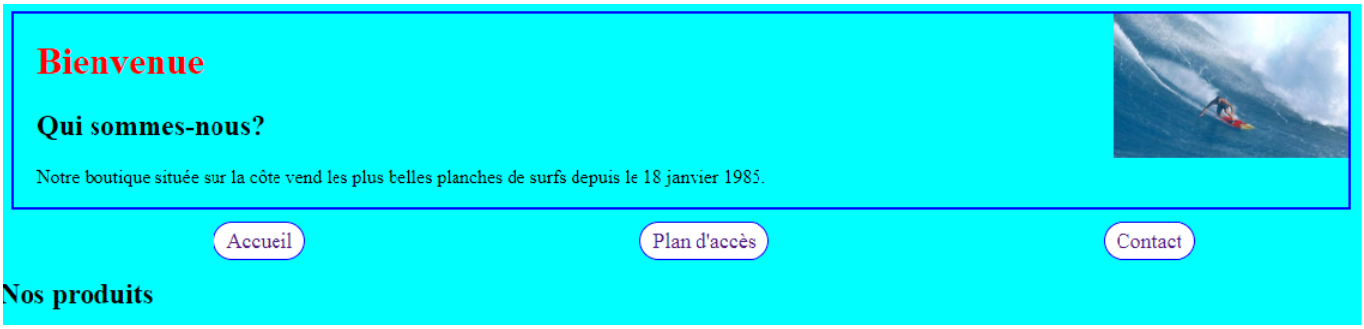
```
.item {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```



Jouer avec les flex

Question 6 : flex

En utilisant le positionnement à base de boites flexibles et quelques décorations, faire que le menu s'affiche horizontalement tel que :



En déclarant :

a. La zone ul du menu en « `flex` »,

b. Chaque item du menu occupant un tiers de l'espace disponible.

c. Le lien de chaque item sans décoration, avec un fond blanc, une bordure d'1 pixel arrondie, une marge intérieure de 10 pixels à gauche et à droite, de 5 pixels en haut et en bas et une taille de police large.

d. Positionner les blocs de produits et les éléments du bas de page les uns à côté des autres afin d'obtenir le résultat suivant :